GETS32 Time Stamps

Related problems

Wed, Aug 8, 2007

The `GETS32` protocol specifies three time stamps that are to be used in the reply header. The Cycle time stamp relates to the most recent `0x0F` event occurrence that occurs about 17 ms ahead of the Booster reset event for the current 15 Hz cycle. The Collection time stamp relates to the time that the data is digitized; but for the special case of a request based upon an event+delay, it relates to the (event+delay) time. The Reply time stamp merely relates to the time the reply message is prepared for network delivery. It is expected that all supporters of the protocol will agree exactly on the Cycle time stamp for all replies. In addition, there will be agreement on the Collection time stamp for the case of (event+delay) requests. This note explains why this last requirement is a problem for our front ends.

The front ends operate synchronously with the 15 Hz accelerator, executing a series of tasks initiated by an interrupt by a timer set for a (15 Hz) event plus delay. To be specific, Linac front ends use a 3 ms delay after the `0x0C` clock event, which occurs every 15 Hz cycle at Booster reset event time. The Booster BLM front ends use a 45 ms delay in order to collect new data after beam has been accelerated and extracted from the Booster, by which time everything is known about that cycle.

Replies are sent from front ends as soon as they are ready. Each cycle, at the start of the `Update` task that begins the 15 Hz series of activities, data is read from whatever hardware interfaces are used and stored into the local 15 Hz data pool. Local applications are run to use this fresh data and possibly to modify or add to the data pool. Finally, replies are built for all active data requests for which a reply is due on the current cycle. Maintaining this sequence of activities each cycle within the `Update` task ensures that a data request received from the network cannot “see” a partially-updated data pool. To an outside user, the data pool seems to be updated at once. This ensures that all values from the data pool included in a reply message is correlated; *i.e.*, it was all updated on the same accelerator cycle. This correlation scheme has been maintained from the earliest days of Linac controls software support.

In the Linac control system, most devices are defined for Acnet so that a single node (`LIP600`) serves as the source node. We refer to this node as a server node that helps to ensure that Linac data, even that gathered by the server from multiple contributing nodes, arrives at the requesting node together. The scheduling is such that the server node delivers the replies to the requesting nodes at 40 ms after the start of its own 15 Hz activities; thus the delivery time is a deadline by which time all contributing front ends must have delivered their replies that are due on the current cycle. Current operation diagnostics for Linac nodes show that the server node has received all of these replies for a given cycle by about 26 ms past the start of its cycle. (A big reason for this apparent leisurely delivery is that the contributing nodes have to acquire their own data pool data from one or more SRMs that are themselves interfaced via arcnet.)

Return to the question of identical time stamps. The definition of all `GETS32` time stamps is that of calendar time, specifically, the number of milliseconds since the start of the calendar year 1970. This is a number of about 40–41 bits in length, which is comfortably delivered as a 64-bit integer. Today, one can derive such times accurately (within 10 μs) using GPS hardware, or using the `0x8F` clock event that is made to occur at the start of each calendar second combined with calendar time obtained from a network time server. Current `GETS32` logic uses the latter method.

The Cycle time stamp marks the time of the most recent `0x0F` clock event, as stated above. Any node that knows accurate time and has access to a clock event decoder can derive this time stamp. But derived independently, the actual value can be off by one millisecond. The only way to ensure that every node supporting the protocol can produce the same identical value for this time stamp is to receive it from a multicast network message. There is one currently used for the purpose that arrives soon after the `0x0F` clock event. If every node listened to this multicast message, the Cycle time stamps could be made identical.

The Correlation time stamp is only significant in this discussion for a request that is based on an (event+delay). In order for that time stamp to be identical across all nodes, the time-stamped events that occurred since the previous `0x0F` event must be processed. Furthermore, they must be processed in an identical fashion, with identical rounding algorithms, etc. But since the `0x0F` event occurs about 17 ms before the Booster reset event that starts the current cycle, its event list of all events occurring since the previous `0x0F` event cannot include the future Booster reset event. That means that the Correlation time stamp cannot be computed properly by the time the reply message for data from the (event+delay)-indicated cycle is due for delivery. So, even if all nodes listen to the multicast message, they cannot know the required standardized event time stamps that are needed to compute the required correlation time stamp field in the reply header, since it will not be known until soon after the `0x0F` event that presages the *following* cycle. Again, any node that knows accurate calendar time can compute the required time stamp, but it cannot guarantee an identical value; it will sometimes be off by 1 least-significant-bit.

Note that almost all event-based data requests used for Linac data specify an event that occurs at Booster reset event time. It is natural in the world of 15 Hz front ends. For a specified zero delay following the event, such a request is interpreted as asking for the data pool values that are updated early in the imminent cycle, often including sample-and-hold values captured during the 50 μs Linac beam pulse. To sample Linac beam data, say to read it from the hardware, literally at the time as the `0x1D` event, would result in nothing of interest, as Linac beam is not accelerated until 2 ms after that event time.

To summarize, the Cycle time stamps could be made identical across front ends if every front end listened to the multicast message sent following event `0x0F`. But to make the Correlation event time stamps identical seems impractical because the needed clock event time stamps are not known by the time reply messages are built. Indeed, the multicast event message is not useful for Linac front ends because of this. Alternatively, if the `GETS32` time stamps were merely accurate, not necessarily identical, then any node that knows accurate time can participate. In this case, "accurate" could be expected to imply normally identical values that could sometimes be off by 1 millisecond. It would seem this could be accurate enough for sorting out 66 millisecond cycles.